Department of Computer Engineering
CS 550 – Machine Learning
Hw3 – Report

Oğuzhan Çalıkkasap
21801131

16.12.2018

1. Introduction

In a cost sensitive classification task, main idea is to find an ideal feature subset that is cost-light and yet containing sufficient information to classify the data correctly. For this task, I used genetic algorithm to select the appropriate features and SVM to classify them.

2. Algorithm

Pseudo-code of my algorithm is as follows:

1) Determine the encoding scheme (bit representation for each feature)
2) Determine the fitness function
3) Randomly Initialize the population
4) Perform classification with features selected by the initial population
5) Compute the F-beta score and the total feature cost for each DNA in population
6) Evaluate the fitness of DNAs using the F-beta score and total cost. (to be explained in detail)
7) Perform genetic operations selection, crossover and mutation if stopping criterion is not satisfied.
8) Repeat steps 4 to 7 until a subset is found with the following properties: (less than 3/4 of total cost & min 70% class-based & min 85% overall f-beta score) OR the number of generations reach a predetermined threshold.

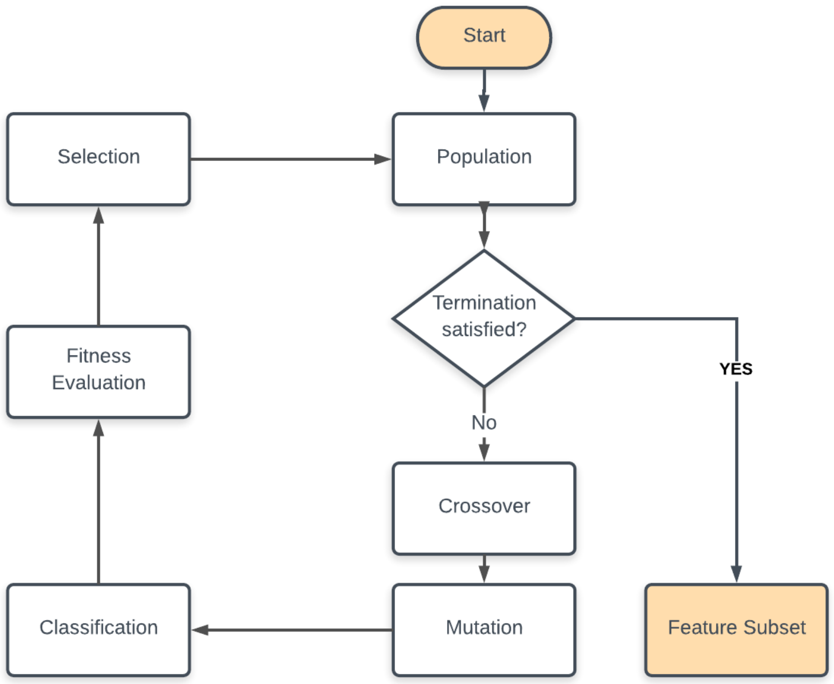


Figure 1: Flow-chart of the algorithm

3. Classifier: Class-weighted Support Vector Machine

<u>Parameters used:</u>

C = 6, **kernel** = ‘linear’, **gamma** = ‘auto’, **class_weights**={class1: *97.5*, class2: *94.9*, class3: *7.5*}

I have used SVM as the classifier because of its decent performance in higher-dimensions, simplicity, and versatility where different kernels can be easily adapted for decision functions. Since Thyroid dataset is quite imbalanced, I assigned different weighted penalty coefficients to each class. To do so, I calculated the class occurrence frequencies from the training set, and determined their penalty value between 0-100 proportional to these frequencies. Thanks to the special penalty values, I was able to benefit the fastest ‘linear’ kernel with this parameter settings, achieving over 94% class-based accuracy. Gamma value ‘auto’ is equal to 1 / n_features. While default value of overall penalty value C is 1, I experimentally set it to 6 to obtain greater class-based and overall accuracies.

With the introduction of class-weighted penalty values, classical SVM transforms to the minimization problem as below:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) + C_{pos} \sum_{i \in \mathcal{P}} \xi_i + C_{neg} \sum_{i \in \mathcal{N}} \xi_i, \\ s.\ t. \quad & y_i \Big(\sum_{j=1}^{N} \alpha_j y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) + b \Big) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \end{aligned}$$

where P and N represent the positive training instances. There are multiple C values whereas there is one in standard SVM. Misclassification penalty for minority class is chosen to be larger as mentioned in the parameters above (class_weights).

4. Feature Representation, Genetic Operators and Parameters

In order to represent the presence or absence of each features, I used a bit string which has a length of 21, number of all features. So the following DNA example represents the presence of second and last three features, while the rest remains absent: 010000000000000000111.

4.1. Selection

Parameter: **r** = 0.4, where selection ratio = 1-r/p. These DNAs are remains in the next generation. Roulette-wheel selection method is employed.

4.2. Crossover

Parameter: r*p/2 pairs are selected for crossover. They are probabilistically crossed-over and added to the next generation. Mask used: *111000111000111000111*

4.3. Mutation

Parameter: **m** = 0.3, where mutation ratio = m*p. They are selected from the next generation obtained in result of above genetic operations. Point mutation is used.

4.4. Number of Population = 6

4.5. Number of Generations = 21

5. Fitness Function

I was required to take into account feature extraction and misclassification costs together, therefore I used F-beta score and total cost of selected features. I did not rely on the pure accuracy, since the dataset was highly imbalanced and even choosing class3 for all the time would yield an accuracy of around 91%. To get more reliable evaluation results, I utilized F-beta score, formulized as below:

$$F_\beta = (1 + \beta^2) \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{(\beta^2 \cdot \mathrm{precision}) + \mathrm{recall}}$$

As seen in the formula, F-beta score is the weighted harmonic mean of precision and recall, reaching its optimal value at 1 and its worst value at 0. The *beta* parameter determines the weight of precision in the combined score. beta < 1 lends more weight to precision, while beta > 1 favors recall. I emphasized Recall with assigning beta=1.5 for this classification task, giving more importance to classification of minority classes. I started to road by using F-beta with the cost in the following format. The idea is to get a higher fitness value when F-beta score is higher whereas the feature extraction cost is lower:

$$Fitness = \frac{F - beta\ score}{Selected\ feature\ subsett\ cost}$$

Even if it seems logical, it did not give the results I expected by ending up classifying all instances as class3 (majority), and I figured out why with some further inspection. It was simply because the scale of change in F-beta scores were not comparable to the change in feature costs. F-beta score can take values between 0 and 1, whereas the total selected feature cost varies between minimum 0 (meaning no features) and 102.3 (all features) maximum. In order to tackle this problem, I introduced a proper modification to the presented formula and my final fitness function has become:

$$Fitness = \frac{e^{7*F\beta}}{Feature\ costs}$$

Range of the fitness value: [0.01, 1097]

Multiplying the F-beta score with seven chosen experimentally to make its change more effective in the fitness value. I also used exponential function to further enhance the effect of relatively small improvements in the greater accuracy interval. In other words, improving accuracy in the 85%-100% range gives more prize compared to an improvement in <85% interval.

6. Results

After constructing my SVM classifier with the mentioned parameters, I ran genetic algorithm on train set to select features wisely. At the end, I obtained last generation's fittest DNA (selected features), and have my SVM classifier perform predictions on test set using the feature subset determined by the DNA. Results are reported below:

6.1 Fittest DNA of Last Generation and Selected Features:

111001100010010111000 with fitness value: *119.19*

[age, sex, on_thyroxine, sick, pregnant, query_hyperthyroid, hypopituitary, psych, TSH, T3]

6.2 Total Cost of Selected Features:

1+1+1+1+1+1+1+1+22.78+11.41 = *42.19*

6.3. Train Set Results:

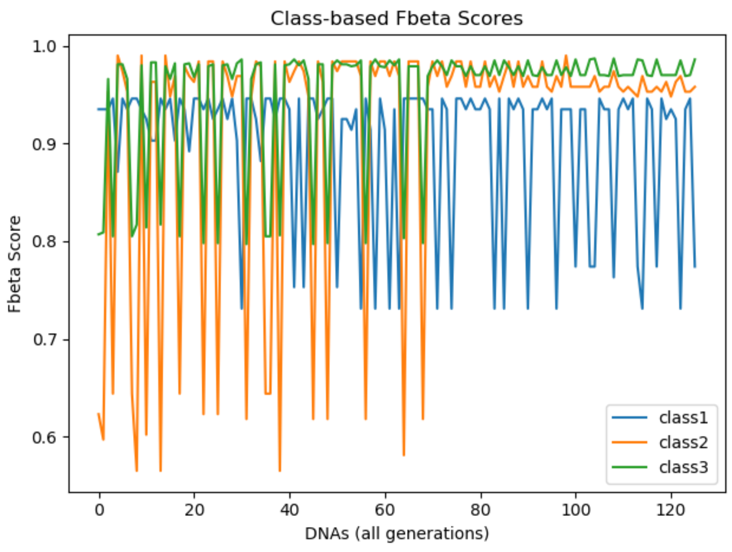
Class-based accuracies: *[0.794 0.958 0.986]* F-beta: *0.94*

6.3 Test Set Results and Graphs:

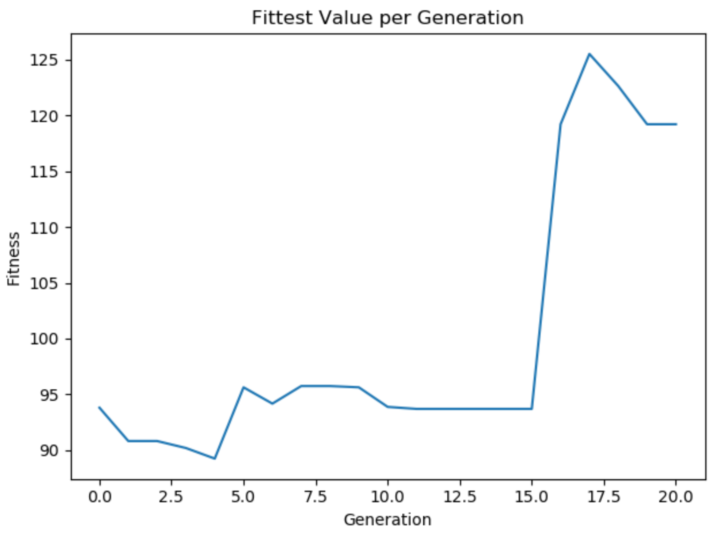
Overall accuracy: *0.971*, Class-based: *[0.709, 0.96 , 0.978]* Confusion matrix:

```
[  51,   22,    0],
[   7,  170,    0],
[  18,   53, 3107]
```

6.4. Class-based Accuracies during Evolution:



6.5. Fittest Value per Generation:


Fittest Value per Generation
Fitness
Generation